REMARKS

The examiner objected to claim 2 on the ground that the recitation "optional_token" is a label having no grammatical meaning. In response, applicant amended claim 2 to recite features similar to the features previously recited in claims 5 and 6. Applicant canceled claims 5 and 6.

The examiner rejected claims 1, 7-9 and 13-15 under 35 U.S.C. §102(e) as being anticipated by U.S. Patent No. 6,314,510 to Saulsbury. The examiner also rejected claims 1, 7, 10 and 13 under 35 U.S.C. §102(e) as being anticipated by Steven's "ALU Design and Processor Branch Architecture," *Microprocessing and Microprogramming*, 1993. Further, the examiner rejected claims 2-6 under 35 U.S.C. §103(a) as being unpatentable over Saulsbury in view of U.S. Patent No. 6,564,316 to Peters. The examiner also rejected claims 10-12 under 35 U.S.C. §103(a) as being unpatentable over Saulsbury.

Specifically, regarding claim 1, the examiner stated:

> 9. Referring to claim 1, Saulsbury has taught a computer program product residing on a computer readable storage medium (Fig. 1, component 102; column 2, lines 42-45) comprising instructions (Fig 4), including a context branch instruction (Fig.4 and column 4, lines 44-45; note the "branch on zero" (bz) instruction) that, when executed, causes a data processing apparatus to select another instruction in an instruction stream from one of a branch target instruction associated with a label specified by the context branch instruction (Fig.4; note that the "bz nextI" instruction will branch to the target instruction associated with label "nextI", which is the "btst wrl, dbl" instruction) and an instruction following the context branch instruction ("st wrO, [addr2]") based on a comparison of a current executing context number to a context number specified by the context branch instruction. The bz instruction is a branch on zero instruction, which means that if the context number specified by the branch instruction (zero) matches a current executing context number (i.e., the value to be compared to zero), then a branch will occur. In this program, the "bz nextI" instruction checks to see if the current executing context number (which may be interpreted as either dirty bit dbO or the flag which would be checked by the bz instruction) is equal to 0, the specified context number. See column 4, lines 39-45. It should be noted that the branch is a context branch and the numbers are context numbers as the branches and numbers are associated with executing contexts (i.e., environments). See the title, abstract, and column 2, lines 40-41.
> b) retrieving the selected other instruction. As is known, with a condition branch, as shown in Fig.4, either the target instruction or the subsequent instruction will be retrieved. (Final Action, pages 4-5, paragraph 9)

Additionally, responding to the applicant arguments presented in the Amendment in Reply to Action of February 7, 2007, the examiner stated:

> 34. Applicant argues the novelty/rejection of claim 1 on page 9 of the remarks, in substance that:
>
> > "Moreover, such a comparison operation would compare the content of two registers and not the current executing context number, as provided by the microprocessor, to a context number value specified by the branch instruction."
>
> 35. These arguments are not found persuasive for the following reasons: a) The examiner asserts that applicant is reading "current executing context number" too narrowly. As the examiner stated, any number in the machine can be considered a context number, as a context is essentially an execution environment, and all values produced by a microprocessor are done so within an execution environment. Therefore, the numbers compared are context numbers. (Final Action, pages 15, paragraphs34-35)

Applicant amended independent claim 1 to clarify that the branch instruction specifies a "thread number" and that execution of the context branch instruction causes a comparison of the executing thread number and the thread number specified by the branch instruction. Support for this clarification is provided throughout the application including, for example, at page 4, lines 4-7, page 6, lines 15-16, etc., of PCT publication No. WO 01/018646. Applicant similarly amended independent claims 7, 10 and 13. As explained in the application, "[t]he hardware-based multithreaded processor 12 has multiple microengines 22 each with multiple hardware controlled threads that can be simultaneously active and independently work on a task" (PCT publication No. WO 01/018646, page 2, lines 12-14.)

Additionally, applicant amended claims 2, 4, 8-9, 11-12 and 14-15 to make the language recited therein consistent with the amended language of the respective independent claims.

Applicant's independent claim 1 recites "a context branch instruction that, when executed, causes a data processing apparatus to: select another instruction in an instruction stream from one of a branch target instruction associated with a label specified by the context branch instruction and an instruction following the context branch instruction based on a comparison of a current executing thread number to a thread number specified by the context branch instruction; and retrieve the selected other instruction." Applicant's context branch

instruction causes the performance of a comparison operation in which the thread number of the currently executing thread (i.e., the value identifying the executing thread) in the data processing apparatus on which the instruction is executed, and the thread number specified by the branch instruction. Execution of the context branch instruction also results in the retrieval of the next instruction to be executed.

In contrast, and as explained in applicant's Amendment in Reply to Action of February 7, 2007, Saulsbury describes a microprocessor that has a reduced context switching overhead for handling traps (col. 1, lines 6-8). To that end, Saulsbury describes a program and a trap handling routine. Specifically, Saulsbury explains:

> **After the modified nop instruction is executed, a trap may occur. In this case, the microprocessor 100 transfers execution from the program to a conventional trap handling routine. The write special register instruction wrspr in the trap handling routine causes the operand stored by the working register wr1 to be saved in a special register sp0 of the special register file 112. The write dirty bit registers instruction wrdbr then causes the dirty bits stored by the dirty bit registers dbr0 to dbrN-1 to be stored in the working register wr1. Once this has been done, the bit test instruction btst determines if the dirty bit stored by the dirty bit register dbr0 is zero or one. If the dirty bit is zero, this indicates that the operand stored by the working register wr0 is inactive. In this case, the trap handling routine branches to the bit test instruction btst at the label next1 as a result of the branch on zero instruction bz. Thus, the operand stored by the working register wr0 is not saved to the main memory 104 because it has become inactive. But, if the dirty bit is one, then this indicates that the operand is active. In this case, the trap handling routine does not branch and the store instruction st causes the operand stored in the working register wr0 to be saved in the main memory 104 at the address addr2. (emphasis added, col. 4, lines 30-51)**

Thus, Saulsbury's processor performs a comparison operation in which the content of two registers (e.g., wr1 and db0) are compared. At no point does Saulsbury disclose that a comparison of the currently executing thread value (as indicated by the processor) is compared to a thread number specified by the branch instruction.

Accordingly, Saulsbury fails to disclose or suggest at least "a context branch instruction that, when executed, causes a data processing apparatus to: select another instruction in an instruction stream from one of a branch target instruction associated with a label specified by the context branch instruction and a sequential instruction following the context branch instruction based on a comparison of a current thread number provided by the data processing apparatus and

a thread number specified by the context branch instruction," as required by applicant's independent claim 1.

As noted, the examiner rejected independent claim 1 as being anticipated by Steven. Specifically, the examiner stated:

> 18. Referring to claims 1, 7, and 13, Steven has taught a computer program product residing on a computer readable storage medium comprising instructions (this is inherent as instructions must be stored somewhere), including a context branch instruction (see page 268, section 4.4, and note the Bcc instruction) that, when executed, causes a data processing apparatus to select another instruction in an instruction stream from one of a branch target instruction associated with a label specified by the context branch instruction (page 268; note that the Bcc instruction will branch to the target instruction associated with the label) and an instruction following the context branch instruction (the fall-through instruction, which inherent exists in conditional branching) based on a comparison of a current executing context number to a context number specified by the context branch instruction. The Bcc instruction compares a register value (specified context number) to a current executing context number (either another register value or immediate specified by the branch instruction). It should be noted that the branch is a context branch and the numbers are context numbers as the branches and numbers are associated with executing contexts (i.e., environments). b) retrieving the selected other instruction. As is known, with a condition branch such as the Bcc instruction, either the target instruction or the subsequent instruction will be retrieved. (Final Action, pages 7-8, paragraph 18)

As described in Steven:

> ***4.4. Combined compare and branch instruction***
>
> **The MIPS [15] and MIPS-X [6] RISC processors, along with their commercial derivatives [19] break completely away from the traditional condition code model by providing combined compare and branch instructions with the following format:**
>
> **Bcc Rsrc1, src2, label**
>
> **If the specified relationship holds between two registers or a single register and an immediate value then the branch is taken. (Steven, page 268, left column)**

Thus, Steven's combined compare and branch instruction compares at least one register to either another register or to an immediately value. However, Steven neither describes nor suggests comparing the thread number of a currently executing thread on a processor, with a thread number specified by the branch instruction. Accordingly, Steven also fails to disclose or

suggest at least the features of "a context branch instruction that, when executed, causes a data processing apparatus to: select another instruction in an instruction stream from one of a branch target instruction associated with a label specified by the context branch instruction and a sequential instruction following the context branch instruction based on a comparison of a current thread number provided by the data processing apparatus and a thread number specified by the context branch instruction," as required by applicant's independent claim 1.

Because none of the prior art references cited by the examiner to reject claim 1 discloses or suggests, alone or in combination, at least the features of "a context branch instruction that, when executed, causes a data processing apparatus to: select another instruction in an instruction stream from one of a branch target instruction associated with a label specified by the context branch instruction and a sequential instruction following the context branch instruction based on a comparison of a current thread number provided by the data processing apparatus and a thread number specified by the context branch instruction," applicant's independent claim 1, and the claims that depend from it, are therefore patentable over the cited art.

Independent claims 7, 10 and 13 recite "performing a comparison of a thread number of an executing thread to a thread number specified by a context branch instruction; selecting another instruction in an instruction stream from one of a branch target instruction associated with a label specified by the context branch instruction and an instruction following the context branch instruction based on the comparison," or similar language. For reasons similar to those provided with respect to independent claim 1, at least these features are not disclosed by the cited art. Applicant's independent claims 7, 10 and 13, and the claims that respectively depend from them, are therefore patentable over the cited art.

All of the dependent claims are patentable for at least the reasons for which the claims on which they depend are patentable.

In view of the foregoing, applicant respectfully submits that the application is in condition for allowance and such action is respectfully requested at the examiner's earliest convenience.

Canceled claims, if any, have been canceled without prejudice or disclaimer.

Any circumstance in which the applicant has (a) addressed certain comments of the examiner does not mean that the applicant concedes other comments of the examiner, (b) made

arguments for the patentability of some claims does not mean that there are not other good reasons for patentability of those claims and other claims, or (c) amended or canceled a claim does not mean that the applicant concedes any of the examiner's positions with respect to that claim or other claims.

No fee is believed due. Please apply any other required fees to deposit account 06-1050, referencing the attorney docket number shown above.

Respectfully submitted,

Date: July 9, 2007

Ido Rabinovitch
Attorney for Intel Corporation
Reg. No. L0080

Customer No. 20985
Fish & Richardson P.C.
Telephone: (617) 542-5070
Facsimile: (617) 542-8906